YLDLOCK / ANON – Verify Your Download & Vault Release Model

1. Verify Your Download

Download verification is optional, but highly recommended. These steps ensure the installer you run is the exact one we published – cryptographically anchored to the same settlement route as every other verified user – and not a tampered copy.

Verification Process Overview

1.Download the YLDLOCK / ANON installer for your platform
2.Download the hash list `YLDLOCK-SHA256SUMS`
3.Download the signature bundle `YLDLOCK-SHA256SUMS.asc`
4.Import one or more trusted YLDLOCK release keys
5.Verify the signatures on the hash list
6.Verify that your installer's hash appears in that list

If all checks pass, your installer is cryptographically anchored to the official YLDLOCK release route.

Files Used in Verification

For each release we publish:

Installer / Binary

- `yldlock-vault-1.0.0-win64-setup.exe`
- `yldlock-vault-1.0.0-x86_64-apple-darwin.zip`
- `yldlock-vault-1.0.0-x86_64-linux.tar.gz`

Hash List – `YLDLOCK-SHA256SUMS`

A plain-text list of SHA-256 hashes for all official build artifacts.

Hash Signatures – `YLDLOCK-SHA256SUMS.asc`

A bundle of OpenPGP signatures over the hash list, created by multiple independent YLDLOCK builder keys (security, infrastructure, release engineering).

Note: You only need to trust one builder key, but you're free to require more.

---

2. Import YLDLOCK Release Keys (All Platforms)

Before you verify anything, load one or more YLDLOCK release keys into GPG.

Import Individual Key

```bash
gpg --import yldlock-builder-ops.gpg
```

Import All Keys

```bash
git clone https://github.com/yldlock/builder-keys.git
gpg --import builder-keys/*
```

Refresh Keys

```bash
gpg --keyserver hkps://keys.openpgp.org --refresh-keys
```

---

3. Verify on Windows (PowerShell)

Assumptions

•Downloads in `C:\Users\<You>\Downloads`
•Installer: `yldlock-vault-1.0.0-win64-setup.exe`

•Gpg4win (GnuPG) installed

Step 3.1 – Change into your download folder

```powershell
cd $env:USERPROFILE\Downloads
```

Step 3.2 – Check the SHA-256 hash

```powershell
Get-FileHash .\yldlock-vault-1.0.0-win64-setup.exe -Algorithm SHA256
```

Compare the Hash output against the corresponding line in `YLDLOCK-SHA256SUMS`:

```powershell
Get-Content .\YLDLOCK-SHA256SUMS
```

Step 3.3 – Verify the signatures on the hash list

```powershell
"C:\Program Files (x86)\GnuPG\bin\gpg.exe" --verify YLDLOCK-SHA256SUMS.asc
```

If:

•The signatures are valid, AND
•The hash of your installer matches `YLDLOCK-SHA256SUMS`

Then your Windows installer is anchored to the official YLDLOCK release route.

---

4. Verify on macOS

Assumptions

•Downloads in `~/Downloads`
•File: `yldlock-vault-1.0.0-x86_64-apple-darwin.zip`

Step 4.1 – Change into the download directory

```bash
cd ~/Downloads
```

Step 4.2 – Verify the hash is in the hash list

```bash
shasum -a 256 --ignore-missing --check YLDLOCK-SHA256SUMS
```

You should see a line ending with `: OK` next to your file name.

Step 4.3 – Verify the signatures on the hash list

```bash
gpg --verify YLDLOCK-SHA256SUMS.asc
```

If both hash and signatures check out, your macOS binary is cryptographically anchored.

---

5. Verify on Linux

Assumptions

•Downloads in `~/Downloads`
•File: `yldlock-vault-1.0.0-x86_64-linux.tar.gz`

Step 5.1 – Change into the download directory

```bash
cd ~/Downloads
```

Step 5.2 – Verify the hash is in the hash list

```bash
sha256sum --ignore-missing --check YLDLOCK-SHA256SUMS
```

Step 5.3 – Verify the signatures

```bash
gpg --verify YLDLOCK-SHA256SUMS.asc
```

When both checks pass, your Linux tarball is anchored to the verified YLDLOCK settlement route.

---

6. Extra Assurance: Reproducible Builds & Multi-Signer Anchoring

Reproducible Builds

Anyone can build YLDLOCK Vault from source and confirm the resulting hashes match those in `YLDLOCK-SHA256SUMS`.

Independent Builders

Multiple YLDLOCK contributors build Vault independently, then each signs the shared `YLDLOCK-SHA256SUMS`. `YLDLOCK-SHA256SUMS.asc` is the stack of those signatures.

Anchored Settlement Route

When you verify the signatures on `YLDLOCK-SHA256SUMS` and the hash of your installer, you anchor your binary to the same route as all other verified clients.

YLDLOCK Vault – Institutional / Industrial Grade Notes

Release & Verification Model

Multi-Party Signed Releases

•Each hash list (`YLDLOCK-SHA256SUMS`) is signed by multiple independent hardware-backed keys (Security Ops, Infrastructure, Core Protocol).
•Signatures are distributed as `YLDLOCK-SHA256SUMS.asc`, enabling custom institutional trust policies (e.g., "require 2 of 3 signers").

Deterministic Build Pipeline

•Vault binaries are produced via a deterministic build system modeled on reproducible builds.
•Clients and regulators can rebuild from source to obtain identical hashes.

Public Builder Key Registry

•YLDLOCK exposes a public builder key registry with:

•Key fingerprints and identities
•Role and validity windows (activation/retirement)
•Institutions can pin specific keys and enforce this via CI/CD.

Independent Verification Paths

•Retail users follow the website "Verify Your Download" guide.
•Institutions integrate hash + signature verification into their deployment pipelines on Linux, macOS, and Windows.

Cryptographic Architecture

PSBT-First Transaction Flow

•Vault treats every spend as a PSBT, keeping signing devices and online coordinators separated.
•Supports Creator / Updater / Combiner / Finalizer roles so institutions can insert their own policy engines, HSMs, or manual approvals.

Rotating Receive Routes

•Deposit addresses / pubkeys rotate per invoice or per session to reduce on-chain correlation.
•Internally, YLDLOCK maps all routes deterministically to the same Vault account.

Anchored Settlement Routes

•Deposits are treated as "anchored" when:

1.They reach the configured number of on-chain confirmations.
2.They are linked to the Vault's settlement graph and visible in the dashboard.

Non-Custodial Signing Options

•Institutions can keep keys in external signers (HSMs, hardware wallets, offline nodes).
•YLDLOCK never needs raw private keys; it coordinates using descriptors and signed PSBTs.

Operational & Compliance Notes

Software Distribution Integrity

•Public installers and connectors are shipped with SHA-256 hashes and multi-signer PGP signatures.
•Verification commands are documented per operating system and can be automated.

Key Management & Rotation

•Release signing keys are:

•Stored on hardware tokens or HSMs.
•Protected by multi-party signing procedures.
•Rotated on schedule or on security events.

Auditability

•For each release, YLDLOCK can supply:

•Commit ID / tag
•Build metadata (toolchain versions, environment)
•Full hash sets and signatures
•Auditors can replay builds and verify that shipped binaries match the open-source reference.

Separation of Duties

•Clear separation between build, review, and release processes.

Privacy-Preserving Account Model

•Rotating receive routes reduce on-chain linkage while preserving internal deterministic mapping for reconciliation and reporting.

Network-Level Resilience

•Vault connects to multiple settlement backends (full nodes, L2s).
•Institutional customers can point Vault to their own node and observability stack.